IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Guthrie et al.	Examiner:	Ho, Andy
Serial No.:	09/894,828	Group Art Unit:	2194
Filed:	06/29/2001	Docket:	40062.250US01
Confirmation No.:	3720	Notice of Allow. Date:	n/a
Due Date:	02/17/2006		
Title:	ASP.NET HTTP RUNTIME		

CERTIFICATE UNDER 37 CFR 1.10:
"Express Mail" mailing label number: EV 727944285 US
Date of Deposit: February 17, 2006

I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

By: _____
Name: Jenifer Week

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

27488
PATENT TRADEMARK OFFICE

Sir:

We are transmitting herewith the attached:

☒ Transmittal Sheet in duplicate containing Certificate of Mailing
☒ Other: Appellant's Supplemental Reply Brief in triplicate
☒ Return postcard

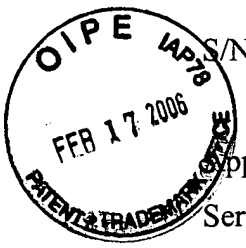
Please consider this a PETITION FOR EXTENSION OF TIME for a sufficient number of months to enter these papers or any future reply, if appropriate. Please charge any additional fees or credit overpayment to Deposit Account No. 13-2725. A duplicate of this sheet is enclosed.

Merchant & Gould P.C.
P.O. Box 2903
Minneapolis, MN 55402-0903
612.332.5300

By: _____
Name: Rene' A. Pereyra
Reg. No. 45,800
RAPereyra/jw

02-21-06

ZMAF



S/N 09/894,828 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Guthrie, et al.	Examiner:	Ho, Andy
Serial No.:	09/894,828	Group Art Unit:	2194
Filed:	June 29, 2001	Docket No.:	40062.0250US01
Title:	ASP.NET HTTP RUNTIME		

CERTIFICATE UNDER 37 CFR 1.10:
"Express Mail" mailing label number: EV 727944285 US
Date of Deposit: February 17, 2006

I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

By: _____
Name: Jenifer Week

APPELLANT'S SUPPLEMENTAL REPLY BRIEF

Mail Stop: Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Supplemental Reply Brief is presented in response to an Office Action mailed November 17, 2005, reopening prosecution after the filing of an Appeal Brief by the Applicants on August 17, 2005. In the Office Action, the Examiner cites a new reference in rejecting all of the pending claims. Applicants hereby reinstate the Appeal filed August 17, 2005, by submitting this Supplemental Reply Brief addressing the Examiner's new ground of rejection. Applicants do not believe that any fees are due with the filing of this Supplemental Reply Brief; however, the Applicants hereby authorize the Commissioner to charge any necessary, required fees to Deposit Account No. 13-2725.

Applicants reserve the right to request an oral hearing by filing a separate request for an oral hearing with the appropriate fee within two months of the date of the Examiner's Answer in response to this Supplemental Reply Brief.

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. §41.37(c)):

I. REAL PARTY IN INTEREST

II. RELATED APPEALS AND INTERFERENCES

III. STATUS OF CLAIMS

IV. STATUS OF AMENDMENTS

V. SUMMARY OF CLAIMED SUBJECT MATTER

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

VII. ARGUMENT

VIII. CONCLUSION

IX. CLAIMS APPENDIX

X. EVIDENCE APPENDIX (None)

XI. RELATED PROCEEDINGS APPENDIX (None)

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is Microsoft Corporation.

II. RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in this appeal:

☒ There are no such appeals or interferences.

☐ These are as follows:

III. STATUS OF CLAIMS

The status of the claims in this application are:

A. TOTAL NUMBER OF CLAIMS IN APPLICATION: 32

B. STATUS OF ALL THE CLAIMS:

1. Claims cancelled: None
2. Claims withdrawn from consideration but not cancelled: None
3. Claims pending: 1-32
4. Claims allowed: None
5. Claims rejected: 1-32

C. CLAIMS ON APPEAL: Claims 1-32

IV. STATUS OF AMENDMENTS

An amendment after final was filed subsequent to the final rejection, but the Examiner did not enter the amendment for the purposes of appeal.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention is generally directed to methods for caching data for use in dynamically generated web pages. *See, e.g.,* Figures 1 and 4, and page 15, line 15 to page 16, line 2. A concise explanation of each independent claim is provided below:

A. Independent Claim 1

Independent claim 1 generally relates to a computer runtime for handling a Hypertext Transfer Protocol (HTTP) request. *See, e.g.,* Figure 3 and paragraphs [23] and [24]. The runtime includes a context object that logically represents a received HTTP request and that encapsulates at least one property of the received request. *See, e.g.,* Figure 3 and paragraphs [24] and [26]. The runtime also includes an event pipeline corresponding to the context object for processing the context object. *See, e.g.,* Figure 3 and paragraphs [24] and [25]. The event pipeline includes a plurality of request events such that when an event (corresponding to a request event) occurs a call-back is generated. *See, e.g.,* Figure 3 and paragraphs [24] and [25]. The call-back initiates each application and each module that is registered for the request event to process the context object. *See, e.g.,* Figure 3 and paragraphs [26] and [27].

B. Independent Claim 10

Independent Claim 11 generally relates to a method for processing an HTTP request. *See, e.g.,* Figure 3 and paragraphs [23] and [24]. The method includes forming a context object that logically represents a received HTTP request and that encapsulates at least one property of the received request. *See, e.g.,* Figure 3 and paragraphs [24] and [26]. The method further includes forming an event pipeline corresponding to the context object for processing the context object. *See, e.g.,* Figure 3 and paragraphs [24] and [25]. The event pipeline includes a plurality of request events, each request event corresponding to an event. *See, e.g.,* Figure 3 and paragraphs [24] and [25]. The method includes generating a call-back when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event. *See, e.g.,* Figure 3 and paragraphs [24] through [27]. The method further includes initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object. *See, e.g.,* Figure 3 and paragraphs [26] through [27].

C. Independent Claim 22

Independent claim 22 is independent claim 11 rewritten into Beauregard form. Independent claim 22 generally relates to a computer-readable medium having computer-executable instructions for processing a Hypertext Transfer Protocol (HTTP) request. *See, e.g.,* Figure 1 and paragraph [18]. The processing method includes forming a context object that logically represents a received HTTP request and that encapsulates at least one property of the received request. *See, e.g.,* Figure 3 and paragraphs [24] and [26]. The processing method further includes forming an event pipeline corresponding to the context object for processing the context object. *See, e.g.,* Figure 3 and paragraphs [24] and [25]. The event pipeline includes a plurality of request events, each request event corresponding to an event. *See, e.g.,* Figure 3 and paragraphs [24] and [25]. The processing method includes generating a call-back when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event. *See, e.g.,* Figure 3 and paragraphs [24] through [27]. The processing method further includes initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object. *See, e.g.,* Figure 3 and paragraphs [26] through [27].

VI. NEW GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The Examiner rejected claims 1-32 under 35 U.S.C. § 103(a) as being unpatentable over Gosling (U.S. Pat. No. 6,247,044) in view of Goward (U.S. Pat. Pub. No. 2002/120677) and Logston (U.S. Pat No. 6,687,735). In order to establish *prima facie* obviousness under 35 U.S.C. 103(a), three basic criteria must be met, namely: (1) there must be some suggestion or motivation to combine the references or modify the reference teaching; (2) there must be a reasonable expectation of success; and (3) the reference or references when combined must teach or suggest each claim limitation (Manual of Patent Examining Procedure 2142). Applicants submit that the Office Action failed to state a *prima facie* case of obviousness, and therefore the burden has not properly shifted to Applicants to present evidence of nonobviousness. Applicants respectfully submit that the Examiner has failed to establish a *prima facie* case of obviousness because the references cited by the Examiner fail to disclose or suggest all of the limitations of the pending claims as discussed herein, and furthermore, the references are not properly combinable.

Specifically, the grounds of rejection presented for review are:

1. With respect to all claims, whether Goward teaches or discloses the claimed element "forming an event pipeline corresponding to the context object."

2. With respect to all claims, whether there is adequate motivation to combine the teachings of Goward with the teachings of Gosling.

3. With respect to all claims, whether Logston teaches or discloses "generating a call-back event." **(Arguments Addressing this Ground were Filed in The Appeal Brief And are Reproduced Below).**

4. With respect to all claims, whether Logston teaches or discloses "initiating each application and each module that is registered in association with the request event." **(Arguments Addressing this Ground were Filed in The Appeal Brief And are Reproduced Below).**

VII. ARGUMENT

The arguments presented below are intended to address the Examiner's new grounds for rejecting claims 1-32 of the present application, presented in the Office Action of November 17, 2005. The new grounds are based on the newly cited Goward reference (hereinafter "Goward"). Although the Examiner indicates that the new grounds of rejection render the arguments filed in the Appeal Brief (August 17, 2005) moot, Applicants respectfully disagree. Specifically, with respect to the Logston reference, the Examiner still relies on this reference as teaching the generating a call-back when the event corresponding to the request event is raised and when at least one of an application and a module is registered in association with the request event and initiating each application and each module that is registered in association with the request event to process the context object, limitations recited in the claims. *See Office Action* (November 17, 2005), page 4. For the convenience of the Board, the Applicants have included below the arguments presented in the Appeal Brief that relate to the Logston reference, and that show that Logston does not disclose these elements of the claims.

A. Claims 1-32

1. GOWARD FAILS TO ANTICIPATE "FORMING AN EVENT PIPELINE CORRESPONDING TO A CONTEXT OBJECT"

The Examiner asserts that Goward teaches a system of using a servlet to service a client request wherein an event pipeline is created such that multiple callbacks are made to the servlet to utilize state information maintained within the servlet. *See Office Action* (November 17, 2005), page 3. For the reasons explained in detail below, Applicants submit that Goward does not disclose, "forming an event pipeline corresponding to a context object."

Goward is directed to a method and apparatus for providing a servlet that interoperates with a server page to implement a web application. *See Goward*, page 1, paragraph [0002]. The method includes executing a server page by compiling the server page and then executing the compiled server page to generate the display page. *See Goward*, page 1, paragraph [0012]. In one embodiment, the execution of the compiled

server page includes performing one or more callbacks to the servlet in order to utilize state information maintained within the servlet. *See Goward*, page 1, paragraph [0012].

Applicants first note that the Examiner has not made clear what features of Goward, the Examiner is equating to an event pipeline and to a context object. However, Applicants submit that by any reading, Goward cannot be reasonably interpreted to disclose "forming an event pipeline corresponding to a context object."

The Detailed Description defines both the event pipeline and the context object. An event pipeline is "an event-based architecture that includes a plurality of request events for generating a callback when the request event is raised." *Detailed Description*, page 9, paragraph [23]. The context object logically represents a received request by encapsulating properties associated with the request and that are needed for processing the request. *Detailed Description*, page 9, paragraph [22]. Thus, as defined by the Detailed Description, the claim limitation at issue includes forming a plurality of request events corresponding to an object that encapsulates properties of a request. Goward does not disclose the formation of a plurality of request events corresponding to an object that encapsulates properties of a request.

It appears to the Applicants that the Examiner is equating the multiple callbacks within a compiled server page, disclosed by Goward, to the event pipeline. However, Applicants point out that Goward does not disclose that the server page is formed corresponding to anything. Rather, the server page is stored in a template and is retrieved by the servlet. *See Goward*, page 3, paragraph [0044] and FIG. 2. For this reason alone, the compiled server page cannot anticipate the event pipeline, because the limitation in the claims is clear that the event pipeline is formed corresponding to a context object.

If the Examiner is arguing that the servlet corresponds to a context object, Applicants kindly note that Goward cannot support such a reading. The servlet cannot be reasonably interpreted as a context object, because the servlet does not encapsulate any properties of a request. As stated by Goward, "web server 109 selects a servlet, such as servlet 110, to execute based upon parameters of the request or other state information (step 404)." *See Goward*, page 2, paragraph [0040]. Similar to the server page, the

servlet already exists on the web server, and is selected by the web server to process a request. Goward does not disclose that the servlet encapsulate features of the request. Furthermore, there is no other element in Goward that can be interpreted as a context object, because Goward never mentions the encapsulation of properties of a request.

Moreover, even if the servlet could be interpreted as a context object, Goward does not disclose that the server page is formed corresponding to the servlet. As previously described, Goward merely teaches that the servlet *retrieves* the server page, which is significantly different than the server page being formed corresponding to the servlet. Thus, for this additional reason, Goward does not anticipate "forming an event pipeline corresponding to a context object."

For at least the reasons stated above, Applicants respectfully submit that Goward does not disclose, "forming an event pipeline corresponding to a context object" as claimed in the pending claims. Accordingly, the Examiner has not established a prima facie case of obviousness. The Applicants kindly request that the Board reverse the Examiner's new ground of rejection, and allow all of the pending claims.

2. GOWARD SPECIFICALLY TEACHES AWAY FROM GOSLING'S SYSTEM

Applicants respectfully submit that even if Goward were to teach the limitation of the claims asserted by the Examiner, Goward cannot be properly combined with Gosling. The Examiner states that the motivation for combining Goward with Gosling is "to allow the system to execute all of the methods defined within the servlet as disclosed by Goward." *Office Action* (November 17, 2005), page 3-page 4. Applicants respectfully submit that someone of ordinary skill in the art would not combine the teachings of Goward with the system of Gosling, as Goward specifically teaches away from systems such as Gosling's.

In the Background section, Goward describes in detail the prior art methods of executing server pages, which utilize servlets executing on a web server. *See Goward*, page 1, paragraph [0006]. With respect to this prior art method, Goward states:

> [0007] Unfortunately, controlling the execution of server pages through servlets can be a complicated task. For example, referring to FIG. 3, a servlet typically first receives a request for a page from a client (step 302). Next, the servlet may compute data that is required for the server page (step 304), and then packages the data into a session (step 306). Next, the servlet performs a redirection to the server page which passes control to the server page and causes the server page to execute (step 308).

Goward, page 1, paragraph [0007]. Moreover, Goward describes that the "programming [for] this type of system is complicated because code must be explicitly included within the servlet to package the data, and code must be explicitly included in the server page to retrieve the packaged data." *Goward*, page 1, paragraphs [0008]. Finally, Goward states that "what is needed is a method and an apparatus for executing server pages without the above-described complications that arise from packaging and unpackaging data." *See Goward*, page 1, paragraph [0009].

Despite these teachings, the Examiner attempts to combine Goward with Gosling, which discloses the exact prior art system from which Goward teaches away. Gosling describes a system that uses a plurality of servlet objects on a server to process requests from a client. *See Gosling*, col. 1, line 60-col. 2, line 4. The servlets are used to compute data that is required to service an internet request. *See Gosling*, col. 3, line 60-col. 4, lines 5. Gosling teaches that the servlets are "executed to obtain dynamically generated information corresponding to the request." *See Gosling*, col. 2, lines 3-5. Moreover, the system includes the excessive amount of code, criticized by Goward, for packaging data from the servlet (*See Gosling*, col. 7, lines 44-55) and retrieving the packaged data (*See Gosli*ng, col. 6 line 67-col. 7, lines 2). Specifically, code must be written to implement the "HttpResponse" object used by the servlet to package data and send the data back to a server. *See Gosling*, col. 7, lines 14-16. Additionally, code must be included in the

servlet API for the method of retrieving the packaged data from the servlet. *See Gosling*, col. 6, line 67-col. 7, line 2. Indeed, the request/response system is a key part of Gosling's invention, as it provides the foundation for the system. *See Gosling* col. 9, lines 61-62 ("The servlets provide a general framework for services built using the request-response paradigm.").

In light of the disclosures of Goward and Gosling, someone of ordinary skill in the art would be guided away from combining Goward with Gosling. Goward's teachings are clear that system's such as Gosling's are complicated and difficult to manage, which is a specific teaching away from such systems. Accordingly, Applicants respectfully submit that the Examiner has improperly combined Goward with Gosling, and thus has not established a *prima facie* case of obviousness, because there is not proper motivation for combining the references.

For all of the reasons stated above, the Applicants submit that the Examiner has not made a *prima facie* case of obviousness with respect to the new ground of rejection. Applicants kindly request that the Board reverse the Examiner's new ground of rejection, and allow all of the pending claims.

ARGUMENTS PRESENTED IN APPEAL BRIEF FILED AUGUST 17, 2005

D. LOGSTON FAILS TO ANTICIPATE "GENERATING" AND "INITIATING" AS CLAIMED

The Examiner asserts that, while Gosling does not disclose or suggest the "generating" or "initiating" elements as claimed, these elements may be found in Logston. The Examiner presents Logston as anticipating the "generating" and "initiating" elements of the claimed invention. The Applicants respectfully disagree.

Logston discloses a distributed application in which a client portion (Logston's DACP) and a server portion (Logston's DASP) form a single application, the benefit of which is to minimize the amount of code transmitted to the client, to minimize the amount of work done at the client, and to allow easy load-balancing at the server by

allowing multiple identical DASPs to be allocated throughout a server farm as necessary. The important structural elements of Logston include that:

- The DACP and DASP are elements of a single application (Col. 7, lines 57-67);
- The DACP is provided by the server to the client as part of the startup process (Col. 8, lines 17-19);
- Every DASP is identical (Col. 8, lines 1-16);
- Each DACP must be assigned (registered) to a DASP for **all** communications from the DACP (Col. 33, lines 21-27);
- Each DASP is also "registered" with the OS to receive requests from the OS (Col. 33, 10-13) as is well known for any application that must interact with the OS;
- A DASP can service multiple DACPs (Col. 8, lines 1-16);
- A DASP remains in existence from the initial assignment to a DACP until either there is a timeout period during which there are no communications received from a DACP or the last DACP terminates (Col. 34, lines 3-51). Thus, Logston's DASPs may operate forever, under a scenario in which new DACPs are assigned to the DASP as old DACPs terminate.

Applicants believe that the citations provided by the Examiner relate only to the proposition that communication from a client (i.e., the DACP) can be directed to the appropriate DASP running on the server and that the DASP can receive requests from the operating system or other global control programs. This is well known in the art and is a fundamental part of the client server model. That Logston uses the same terminology in "registered" and "callback" does not mean that Logston has anticipated the claimed elements.

Applicants believe that Logston does not disclose the "generating" and "initiating" elements of the claimed invention for several reasons. First, the "generating" claim element includes the requirement that the "at least one of an application and a module is registered in association with the request event." Logston's DASP (while capable of receiving communications from any DACP) is registered for specific DACPs to receive all communications from those DACPs. Logston's DASPs are not registered in association with a specific request event that may be raised by specified request.

Furthermore, Logston does not anticipate the "initiating" element in the claimed invention as that element includes the claimed limitation that each application/module

initiated be "registered in association with the request event..." Even at Logston's initial startup, Logston does not initiate a module registered in association with a request event of an event pipeline. Rather, Logston initiates a DASP registered in association with a specified client portion DACP .

The Examiner may be arguing that the DASP's registration with the operating system anticipates the claimed language, but this is also in error because the argument does not take into account that the request events of the claimed invention are part of the event pipeline that correspond to events that may be raised by a HTTP request. The events disclosed in Logston relate only to the operating system or the OpenCable application, neither of which are events that may be raised by a HTTP request when processed by an event pipeline formed for a context object.

In sum, while Logston does disclose registering DASPs for DACPs and registering a DASP with its operating system, Logston does not disclose registering anything in association with a request event, much less a request event associated with an event pipeline formed corresponding to a context object. For at least these reasons, Logston does not teach or disclose the claimed elements of "generating" or "initializing" with all their limitations. Therefore, the Applicants respectfully request that the rejection be withdrawn and all pending claims be allowed.

VIII. CONCLUSION

As described above, Applicants have shown that the Examiner has failed to establish a *prima facie* case of obviousness. Therefore, it is earnestly requested that the Examiner's rejections be reversed, and that all of the pending claims be allowed.

Respectfully submitted,

Dated: February 17, 2006

René A. Pereyra
Attorney Reg. No. 45,800
MERCHANT & GOULD P.C.
P. O. Box 2903
Minneapolis, MN 55402-0903
(303) 357-1651

IX. CLAIMS APPENDIX

The text of the claims involved in the appeal are:

Listing of Claims:

1. (original) A Hypertext Transfer Protocol (HTTP) request handling runtime, comprising:

a context object logically representing an HTTP request that is received at a host application from a client application, the context object encapsulating at least one property associated with the received HTTP request; and

an event pipeline corresponding to the context object, the event pipeline having a plurality of request events, each request event having a corresponding event and generating a call-back when the event corresponding to the request event is raised and when at least one of an application and a module is registered in association with the request event, each call-back initiating each application and each module that is registered in association with the request event to process the context object.

2. (original) The HTTP request handling runtime according to claim 1, wherein the plurality of request events have a deterministic order.

3. (original) The HTTP request handling runtime according to claim 2, wherein at least one of the plurality of request events is a synchronous request event.

4. (original) The HTTP request handling runtime according to claim 2, wherein at least one of the plurality of request events is an asynchronous request event.

5. (original) The HTTP request handling runtime according to claim 2, wherein the plurality of request events further includes at least one request event having a non-deterministic order.

6. (original) The HTTP request handling runtime according to claim 1, wherein the plurality of request events have a non-deterministic order.

7. (original) The HTTP request handling runtime according to claim 6, wherein the plurality of non-deterministic order request events include an error event.

8. (original) The HTTP request handling runtime according to claim 1, wherein a module is registered in association with a plurality of request events.

9. (original) The HTTP request handling runtime according to claim 1, wherein the event pipeline is a separate instance of the event pipeline for each HTTP request that is received at the host application from a client application.

10. (original) The HTTP request handling runtime according to claim 1, wherein HTTP request runtime parses the received HTTP request to form the context object that logically represents the HTTP request.

11. (original) A method for processing a Hypertext Transfer Protocol (HTTP) request, comprising the steps of:

forming a context object that logically represents an HTTP request that is received at a host application from a client application, the context object encapsulating at least one property associated with the received request;

forming an event pipeline corresponding to the context object, the event pipeline having a plurality of request events, and each request event having a corresponding event;

generating a call-back when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event; and

initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object.

12. (original) The method according to claim 11, further comprising a step of registering a module in association with at least one selected request event.

13. (original) The method according to claim 11, further comprising a step of registering a plurality of modules in association with a selected request event.

14. (original) The method according to claim 11, wherein the plurality of request events have a deterministic order.

15. (original) The method according to claim 14, wherein at least one of the plurality of request events is a synchronous request event.

16. (original) The method according to claim 14, wherein at least one of the plurality of request events is an asynchronous request event.

17. (original) The method according to claim 16, wherein the plurality of request events further includes at least one request event having a non-deterministic order.

18. (original) The method according to claim 11, wherein the plurality of request events have a non-deterministic order.

19. (original) The method according to claim 18, wherein the plurality of non-deterministic order request events include an error event.

20. (original) The method according to claim 11, wherein the step of forming the event pipeline corresponding to the context object forms the event pipeline as a separate instance for each HTTP request received at the host application from a client application.

21. (original) The method according to claim 11, wherein the step of forming the context object includes a step of parsing the received HTTP request to form the context object.

22. (original) A computer-readable medium having computer-executable instructions for processing a Hypertext Transfer Protocol (HTTP) request comprising steps of:

forming a context object that logically represents an HTTP request that is received at a host application from a client application, the context object encapsulating at least one property associated with the received request;

forming an event pipeline corresponding to the context object, the event pipeline having a plurality of request events, and each request event having a corresponding event;

generating a call-back event when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event; and

initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object.

23. (original) The computer-readable medium according to claim 22, further comprising a step of registering a module in association with at least one selected request event.

24. (original) The computer-readable medium according to claim 22, further comprising a step of registering a plurality of modules in association with a selected request event.

25. (original) The computer-readable medium according to claim 22, wherein the plurality of request events have a deterministic order.

26. (original) The computer-readable medium according to claim 22, wherein at least one of the plurality of request events is a synchronous request event.

27. (original) The computer-readable medium according to claim 22, wherein at least one of the plurality of request events is an asynchronous request event.

28. (original) The computer-readable medium according to claim 25, wherein the plurality of request events further includes at least one request event having a non-deterministic order.

29. (original) The computer-readable medium according to claim 22, wherein the plurality of request events have a non-deterministic order.

30. (original) The computer-readable medium according to claim 29, wherein the plurality of non-deterministic order request events include an error event.

31. (original) The computer-readable medium according to claim 22, wherein the step of forming the event pipeline corresponding to the context object forms the event pipeline as a separate instance for each HTTP request received at the host application from a client application.

32. (original) The computer-readable medium according to claim 22, wherein the step of forming the context object includes a step of parsing the received HTTP request to form the context object.

X. EVIDENCE APPENDIX

None

XI. RELATED PROCEEDINGS APPENDIX

None